**GeekBrains**

**«ЦИФРОВЫЕ ПРОФЕССИИ. ПРОГРАММИСТ JAVA»**

**ДИПЛОМНЫЙ ПРОЕКТ**

**«АВТОМАТИЗАЦИЯ ОТЧЁТНОСТИ ОРГАНИЗАЦИИ»**

**Маринин Валентин Владимирович**

**Барнаул 2024**

**Дипломный проект: «Автоматизация отчётности организации»**

**Содержание**

|  |  |
| --- | --- |
| Введение | 3 |
| Глава 1 |  |
| 1. Структура приложения | 6 |
| 1.1. Тип приложения и стиль реализации приложения | 6 |
| 1.2. Требования к приложению | 8 |
| 1.3. Схема работы приложения | 9 |
| Глава 2 |  |
| 2. Реализация приложения | 11 |
| 2.1. Инструменты реализации приложения | 11 |
| 2.2. Реализация ReportManager | 12 |
| 2.3. Реализация и строение базы данных | 14 |
| Глава 3 |  |
| 3. Работа с программой | 19 |
| 3.1. Основной модуль, вход в программу | 19 |
| 3.2. Группа административных подпрограмм-отчётов | 21 |
| 3.2.1. Подпрограмма-отчёт «Административный\_пользователь» | 21 |
| 3.2.2. Подпрограмма-отчёт «Административный\_отчёт» | 30 |
| 3.3. Управленческие, бухгалтерские, статистические отчёты | 37 |
| 3.3.1. Отчёты с произвольной структурой | 38 |
| 3.3.2. Отчёты с предопределённой структурой | 44 |
| Глава 4 |  |
| 4. Установка программы и вспомогательная информация | 45 |
| 4.1. Установка программы | 45 |
| 4.2. Вспомогательная информация | 46 |
| Заключение | 48 |
| Список литературы | 49 |
| Приложения | 51 |

**Введение**

Настоящий дипломный проект призван реализовать автоматизацию управления всей отчетностью банка или предприятия (принципиальной разницы нет, но ориентироваться будем на банковские потребности), а также унифицировать работу с отчетностью и продемонстрировать на нескольких примерах работу с приложением.

Предлагаемое к рассмотрению приложение называется «ReportManager». Данное приложение (далее ReportManager) может быть полезно там, где есть необходимость в большом количестве отчетов различной направленности, будь то финансовые отчеты предоставляемые в государственные структуры (обязательная отчетность) или управленческие отчеты для внутреннего использования менеджерами всех уровней. Возможные сферы применения данного приложения: малые и средние коммерческие банки, предприятия торговли (сетевые магазины), промышленные предприятия. Для крупных банков на рынке программного обеспечения существуют множество готовых профессиональных продуктов, которые охватывают все аспекты банковской деятельности, в том числе и отчетность. Применять такое программное обеспечение на небольших банках, во-первых, дорого, а во-вторых, избыточно, так как у средних и малых банков набор предлагаемых клиентам сервисов и банковских продуктов не такой широкий, как у крупных банков, да и количество ежедневных транзакций (финансовых операций) значительно скромнее. Кроме того, у коммерческого приложения для больших банков могут быть и определенные требования к аппаратной части для развёртывания своих приложений, что дополнительно увеличивает стоимость владения таким программным продуктом. Исходя из этих соображений, здесь может пригодиться предлагаемое решение ReportManager. Те же соображения справедливы и для торговых и промышленных предприятий, Доля финансовой отчетности для банковской сферы значительно больше, чем у промышленных и торговых предприятий, для которых характерно преобладание управленческой (аналитической) отчетности, что не отменяет плюсы применения приложения ReportManager характерные для банковской сферы. Кроме того, поддержка и модификация такого приложения обойдётся дешевле, поскольку будет осуществляться собственным отделом IT, в то время как поддержка готового коммерческого решения может потребовать дополнительных финансовых затрат, а модификация может и вовсе не предполагаться.

К плюсам применения приложения ReportManager, помимо более низких, по сравнению с большими профессиональными банковскими приложениями, стоимостей приобретения и владения, можно также отнести и возможность легкого расширения функционала и модификации приложения собственными силами в зависимости от возникших новых потребностей организации. То есть, наличие более гибкого подхода к возникшим новым условиям, а также более короткий промежуток времени для реакции на эти новые условия.

Какие банки можно отнести к средним и малым? Региональные банки, которые имеют несколько филиалов в пределах своего региона или вовсе не имеют филиалов, например, «СибСоцБанк» - региональный банк в Алтайском крае. Также, можно смело отнести к группе малых и средних банков все банки из третьей и четвертой сотен по размеру нетто-активов в списке банков России, который доступен на сайте ***banki.ru.*** На момент написания дипломной работы в списке числилось 328 банков.

Предлагаемое приложение может быть как основным (единственным) источником формирования отчетности организации, так и гибким, легко модифицируемым дополнением, расширением возможностей основной бухгалтерской программы организации, в которой предусмотрен собственный сервис отчетности.

Еще одной целью ReportManager является унификация и стандартизация работы с отчетностью в коммерческом банке, торговом или промышленном предприятии. Единообразный подход к созданию отчетов и работе со всеми отчётами в рамках приложения ReportManager даст возможность сократить время на обучение нового персонала и освоение новых видов отчетов уже работающими сотрудниками.

При использовании данного приложения предполагается тесное взаимодействие между отделом IT организации и отделом бухгалтерии, поскольку разработкой, внедрением и отладкой новых отчетов будут заниматься сотрудники IT отдела, заказчиками и потребителями здесь будет выступать бухгалтерия и менеджмент организации.

Таким образом, целью проекта является создание приложения ReportManager и описание структуры приложения, теоретическое обоснование и техническое описание этой структуры, описание методов реализации этой структуры с указанием используемых инструментов.

Проект выполняется по специализации «программист Java».

При реализации проекта был использован мой опыт работы в банковской сфере. Опыт составляет более девяти лет и включает в себя работу в отделе аналитики (на позициях от стажёра до главного специалиста), где я занимался банковской отчётностью около трёх лет, далее перешел в отдел автоматизации отчётности банка на позицию начальника отдела автоматизации банковской отчетности и главной книги банка.

Поскольку проект реализован исключительно мной, то все роли в проекте исполнял сам: архитектор, разработчик, тестировщик, GUI-дизайнер.

**Глава 1.**

1. Структура приложения.

1.1. Тип приложения и стиль реализации приложения,

Сегодня за создание оптимальной, теоретически обоснованной структуры приложения призвана отвечать такая область информационных технологий, как «Архитектура программного обеспечения».

Основополагающей идеей дисциплины программной архитектуры является идея снижения сложности системы путём абстракции и разграничения полномочий. На сегодняшний день до сих пор нет согласия в отношении чёткого определения термина «архитектура программного обеспечения». Сайт [Института программной инженерии](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B9_%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B8%D0%B8) Карнеги-Меллон ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Software Engineering Institute, SEI) приводит более 150 определений этого понятия, например такое (из книги «Software Architecture in Practice» (2nd edition), Bass, Clements, Kazman; AddisonWesley 2003): «The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them» («Программная архитектура приложения или вычислительной системы - это структура или составные части системы, которые включают программные элементы, видимые извне свойства этих элементов и взаимосвязи между ними»). Или такое: «Архитектура ПО**-** это искусство и наука строить и проектировать программное обеспечение таким образом, чтобы оно удовлетворяло всем заявленным к нему требованиям, а также обеспечивало максимальную простоту доработки, развертывания и масштабирования приложения». Приложение было написано исходя из этого определения, а также опираясь на это определение и будет сделано описание приложения ReportManager.

Тип реализуемого приложения – «desktop», что означает «настольное приложение» - это программа, запускаемая в виде обыкновенного исполняемого файла на устройстве пользователя. В качестве такого устройства может выступать компьютер, планшет или смартфон. В нашем случае это или стационарный компьютер, или ноутбук.

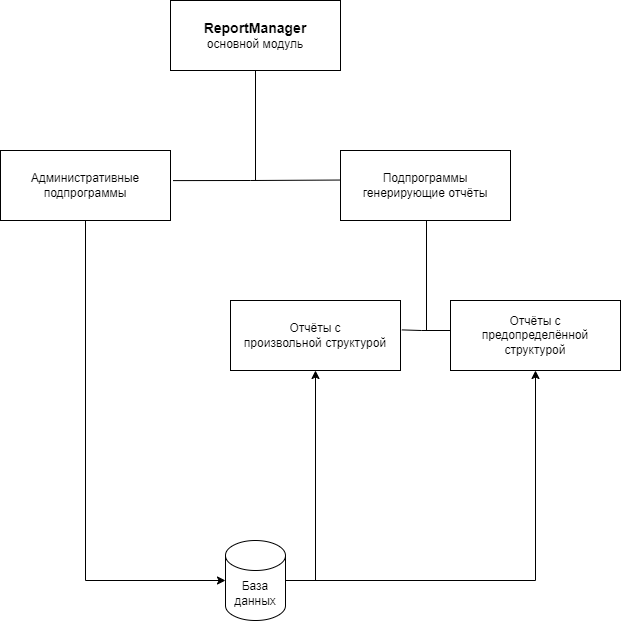
Архитектурные стили - это многократно используемые "пакеты" проектных решений и ограничений, которые применяются к архитектуре для создания выбранных желаемых качеств. Существует множество признанных архитектурных шаблонов и стилей, среди которых:

* [Blackboard](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Blackboard_(design_pattern))
* [Клиент-сервер](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Client%E2%80%93server_model) (этот стиль используют 2-уровневые,  [3-уровневые](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Three-tier_(computing)),  [n-уровневые](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/N-tier) [облачные вычисления](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Cloud_computing))
* Архитектура [основанная на компонентах](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Software_componentry)
* Архитектура [ориентированная на данные](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Database-centric_architecture)
* [Управляемый событиями](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Event-driven_architecture) (или [неявный вызов](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Implicit_invocation))
* [Многоуровневая](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Abstraction_(computer_science)#Layered_architecture) архитектура
* Архитектура микросервисов
* [Монолитное приложение](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Monolithic_application)
* [Одноранговая сеть](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Peer-to-peer) (P2P)
* [Каналы и фильтры](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Pipes_and_filters)
* [Подключаемые модули](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Plug-in_(computing))
* [Реактивная архитектура](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/w/index.php?title=Reactive_architecture&action=edit&redlink=1)
* [Передача репрезентативного состояния](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Representational_state_transfer) (REST)
* [Основанная на правилах](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Rule-based_system) архитектура
* [Сервис-ориентированн](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Service-oriented_architecture)ая архитектура
* [Архитектура без общего доступа](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Shared_nothing_architecture)
* [Архитектура на основе пространства](https://translated.turbopages.org/proxy_u/en-ru.ru.face450c-64b3cbb3-90bf5e99-74722d776562/https/en.wikipedia.org/wiki/Space-based_architecture)

Некоторые специалисты рассматривают архитектурные шаблоны и архитектурные стили как одно и то же, другие рассматривают стили как специализацию шаблонов. Что их объединяет, так это то, что шаблоны и стили являются идиомами для использования архитекторами, они "обеспечивают общий язык" или "словарь", с помощью которого описываются классы систем.

Для реализации приложения ReportManager (основного модуля и подпрограмм-отчётов) использована MVP архитектура. Структура приложения представлена на рисунке 1. Подробнее о структуре в пункте 1.3.

Рисунок 1



1.2. Требования к приложению.

Приложение ReportManager должно быть реализовано таким образом, чтобы, согласно определению архитектуры ПО «обеспечивало максимальную простоту доработки, развертывания и масштабирования». Кроме того, оно также должно иметь простой интерфейс и низкую стоимость владения.

1.3. Схема работы приложения.

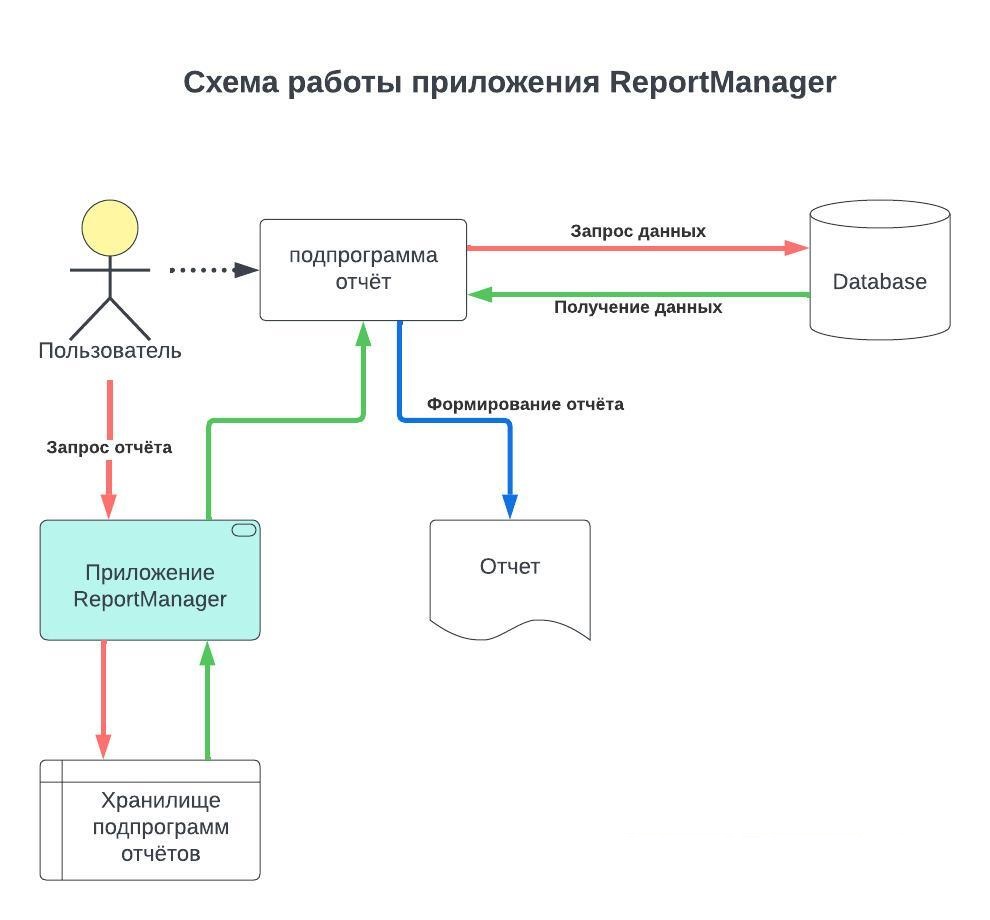
Что представляет собой приложение ReportManager? Основной модуль приложения осуществляет управление и контроль над полным набором отчётов, существующих в банке (или на предприятии). Каждый отдельный отчет представляет собой реализованную в отдельном файле программу (теоретически, эту программу-отчёт можно запустить самостоятельно, без основного модуля), которая вызывается основным модулем ReportManager. Далее, для удобства восприятия, основной модуль называем ReportManager, отдельные программы-отчёты, которыми управляет основной модуль, называем «подпрограмма-отчёт».

Подпрограммы-отчёты делятся на две группы: к первой группе относятся все подпрограммы-отчеты генерирующие отчётность отражающую коммерческую или хозяйственную деятельность банка (предприятия), то есть бухгалтерские, финансовые и управленческие отчеты, вторая группа – это административные подпрограммы-отчёты с помощью которых осуществляется контроль и управление первой группой подпрограмм-отчётов. Под контролем и управлением здесь понимается:

* во-первых, организация доступа конкретного пользователя к определенному перечню отчетов, которые должен генерировать именно этот пользователь, то есть, когда пользователь запускает ReportManager он видит только свои отчёты и больше ничьи;
* во-вторых, контроль версий подпрограмм-отчётов находящихся в хранилище на отдельном сетевом хосте и подпрограмм-отчётов на локальной машине исполнителя, в том случае если версии подпрограммы-отчёта отличаются происходит копирование подпрограммы-отчёта из сетевого хранилища на локальную машину, Такая проверка версий происходит при запуске ReportManager, то есть, осуществляется основным модулем программы. Таким образом, при отсутствии проблем с сетью, у конечного пользователя (исполнителя отчётов) на локальной машине всегда актуальная версия отчёта.
* в-третьих, администратор программы ReportManager (это может быть главный бухгалтер или его заместитель, то есть человек, который распределяет обязанности между сотрудниками и определяет какой сотрудник какие именно отчёты будет генерировать) может давать или отзывать доступ конкретного сотрудника к конкретному отчёту. Набором отчётов, который, как уже указывалось, хранится на сетевом хосте, можно будет манипулировать с помощью отдельной подпрограммы-отчёта из группы административных подпрограмм-отчётов. Более подробно это будет описано далее.

Каждая отдельная подпрограмма-отчёт запрашивает информацию, необходимую для построения отчета, из базы данных. Схему работы приложения демонстрирует Use-case диаграмма, которая приведена на рисунке 1.1.

Рисунок 1.1



Реализация подпрограмм-отчётов осуществлялась, как уже было отмечено, на основе архитектуры MVP. Для понимания детальный картины реализации подпрограмм-отчётов приводятся две структурные UML-диаграммы классов в разделе «Приложения» для одной из административных подпрограмм-отчётов («АДМИНИСТРАТИВНЫЙ\_ПОЛЬЗОВАТЕЛЬ») и подпрограммы-отчёта «Баланс», генерирующей бухгалтерский отчёт. Целью создания диаграммы классов является графическое представление статической структуры декларативных элементов системы (классов, [типов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и т. п.) Она содержит в себе также некоторые элементы поведения (например - операции). При представлении сущностей реального мира разработчику требуется отразить их текущее состояние, их поведение и их взаимные отношения. На каждом этапе осуществляется абстрагирование от маловажных деталей и концепций, которые не относятся к реальности (производительность, [инкапсуляция](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [видимость](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C_(UML)) и т. п.). Классы можно рассматривать с позиции различных уровней. Как правило, их выделяют три основных: аналитический уровень, уровень проектирования и уровень реализации:

* на уровне анализа класс содержит в себе только набросок общих контуров системы и работает как логическая концепция предметной области или программного продукта.
* на уровне проектирования класс отражает основные проектные решения касательно распределения информации и планируемой функциональности, объединяя в себе сведения о состоянии и операциях.
* на уровне реализации класс дорабатывается до такого вида, в каком он максимально удобен для воплощения в выбранной среде разработки; при этом не воспрещается опустить в нём те общие свойства, которые не применяются на выбранном языке программирования.

В нашем случае UML-диаграмма классов представлена проектным уровнем (смотреть раздел «Приложения»).

**Глава 2.**

2.Реализация приложения

2. 1. Инструменты реализации приложения.

Для реализации основного модуля ReportManager и подпрограмм-отчётов использовался язык высокого уровня Java.

Данные для работы приложения (это и данные по коммерческой и хозяйственной деятельности банка или предприятия, и данные для организации работы ReportManager, то есть набор пользователей, отчеты по принадлежности пользователю и т.д.) хранятся в базе данных, при написании дипломного проекта использовалась СУБД MySQL. В реальной жизни выбор СУБД может осуществлять клиент, пользователь приложения ReportManager, исходя из своих финансовых возможностей и любых других предпочтений. Приложение позволяет использовать любую СУБД, поддерживающую стандарт SQL и имеющуюся в наличии у клиента – информация о выбранной СУБД содержится в файле конфигурации, в нашем случае это файл «Config.json». Теоретически, для каждой подпрограммы-отчета можно задать свою СУБД, если в этом возникнет такая необходимость.

2.2. Реализация ReportManager

Всё приложение ReportManager реализовано в виде исполняемых jar-файлов. При старте программы запускается файл основного модуля приложения - ***ReportManager.jar***, который расположен в папке «ReportManager». Эта корневая папка приложения и она включает в себя подпапки:

* Reports – эта подпапка содержит jar-файлы содержащие подпрограммы-отчёты, которые выполняет пользователь. ***ReportManager.jar*** при запуске проверяет наличие доступных пользователю подпрограмм-отчётов (запрос в БД по логину введённому пользователем) и при их отсутствии в этой подпапке (Reports) копирует необходимые jar-файлы с подпрограммами-отчётами из сетевого хранилища в эту подпапку. Также при запуске ***ReportManager.jar*** проверяет версии jar-файлов. Контроль версий реализован на основе расчёта hash-числа для каждого jar-файла и если один и тот же (по названию) jar-файл имеет разные hash-числа, то тогда jar-файл в этой подпапке (Reports) замещается файлом из сетевого хранилища;
* Documents – эта подпапка содержит результат работы подпрограмм-отчётов, то есть собственно готовые отчёты. Сохранение готовых отчётов возможно в текстовом формате, в формате html-документа и в виде файла табличного процессора Excel.

Кроме подпапок в папке «ReportManager» находится файл конфигурации ***Config.json***. Файл содержит следующие ключи:

* "AppHomeDir" – расположение домашней папки приложения, например, «C:/GB/ReportManager/»;
* "ReportsHomeDir" – папка с подпрограммами отчетами (jar-файлами), которая описана выше, тот есть её расположение соответственно будет «C:/GB/ReportManager/Reports/»;
* "ReportsNetDir" – расположение сетевого хранилища со всеми существующими jar-файлами. Туда разработчики выкладывают новые или модифицированные jar-файлы с подпрограммами-отчётами, откуда эти файлы попадут на клиентские машины из-за разницы в hash-числах соответствующих jar-файлов;
* "ReportsResultDir" – расположение папки результирующих отчётов, то есть папки Documents о которой говорилось выше и для нашего примера это соответственно будет «C:/GB/ReportManager/Documents/»;
* "ConfigInfoDB" – строка для подключения к базе данных, например "jdbc:mysql://localhost:3306/enterprise", где «enterprise» это имя базы данных, а вместо localhost должно быть указано имя хоста на котором запущена база данных, в нашем случае база данных - это MySQL;
* "HostDB" – сетевое имя или адрес хоста, где расположена база данных;
* "ConfigFile" – расположение файла конфигурации, значение этого ключа передаётся основным файлом приложения ReportManager.jar всем подпрограммам-отчётам как параметр во время вызова. Таким образом подпрограмма-отчёт получает информацию для подключения к базе данных (ключ «ConfigInfoDB»), а также расположение папки Documents, куда будут сохраняться готовые отчёты.

Помимо файла конфигурации папка «ReportManager» содержит файл ReportManager.bat, что позволяет запустить ReportManager.jar с параметром. А в качестве параметра передаётся расположение и название файла конфигурации. Собственно, можно запустить программу и из командной строки, но для обычного пользователя, на которого рассчитана данная программа (рядовой бухгалтер или менеджер) привыкшего к Windows и графическому интерфейсу, это будет, как минимум, дискомфортно.

2.3. Реализация и строение базы данных.

База данных, как уже упоминалось, реализована на основе СУБД MySQL, версии 8.0.33. На сервере в базе данных «enterprise» хранится вся информация о коммерческой и хозяйственной деятельности организации, которая используется как для построения разнообразных внешних и внутренних отчетов, так и для управления самими отчетами, а также информация для реализации разграничения доступа пользователей к этим отчетам. Поэтому приложение ReportManager можно условно разделить на два блока: административный блок и блок коммерческой и хозяйственной деятельности банка (предприятия). Разделение условное, так как все таблицы с данными обоих блоков хранятся в одной и той же базе данных, в нашем случае это БД «enterprise».

Работа с приложением ReportManager начинается с ввода логина и пароля пользователя которые хранятся в базе данных в таблице «access». Таблица имеет четыре поля:

* уникальный первичный ключ (id);
* внешний ключ, для связи с таблицей пользователей «users» (user\_id);
* поле для хранения логина (login)
* поле для хранения хэша пароля (password)

В поле для хранения пароля «password» находится зашифрованный пароль, для шифрования используется функция хеширования «SHA-256». Пароль, введённый пользователем во время старта программы ReportManager, также шифруется этой функцией и полученный хэш-код сравнивается с табличным, найденным предварительно в БД по логину, указанному пользователем на входе. При совпадении ключей в приложение ReportManager загружается список отчетов, которые доступны этому пользователю для работы.

Регистрацию нового пользователя в приложении ReportManager осуществляет сотрудник предприятия (банка) с правами администратора, запуская соответствующую административную подпрограмму-отчёт из ReportManager. Этот же сотрудник-администратор, используя следующую административную подпрограмму-отчёт. ReportManager даёт доступ новому пользователю к отдельным отчётам из списка всех существующих на предприятии отчётов, которые этот сотрудник должен будет генерировать.

Список всех пользователей (сотрудников предприятия иди банка) содержит таблица «users». Эта таблица включает в себя девять полей:

* уникальный первичный ключ (id);
* фамилия (surname);
* имя (name);
* отчество (patronymic);
* дата рождения (birthday);
* статус актуальности: 1- работает, 2- уволился (status);
* дата трудоустройства (start\_action);
* дата увольнения (end\_action);
* комментарии (comment);

Список всех существующих на предприятии отчётов хранится в таблице «reports». Таблица имеет восемь полей:

* уникальный первичный ключ (id);
* пользовательское название отчета (repname);
* статус актуальности – отчёт действующий или архивный (status);
* дата начала работы отчёта (start\_date);
* дата окончания работы отчёта (end\_date);
* имя подпрограммы реализующей данный отчёт (report\_prog);
* хэш-код подпрограммы (hash\_prog);
* комментарий (comment);

Информация о соответствии пользователей и доступных им отчётов хранится в таблице «user\_reports». Эта же таблица одновременно хранит и историю работы пользователей с тем или иным отчетом – таблица содержит поле-статус актуальности, дату старта и дату окончания работы пользователя с отчетом. Таблица содержит шесть полей:

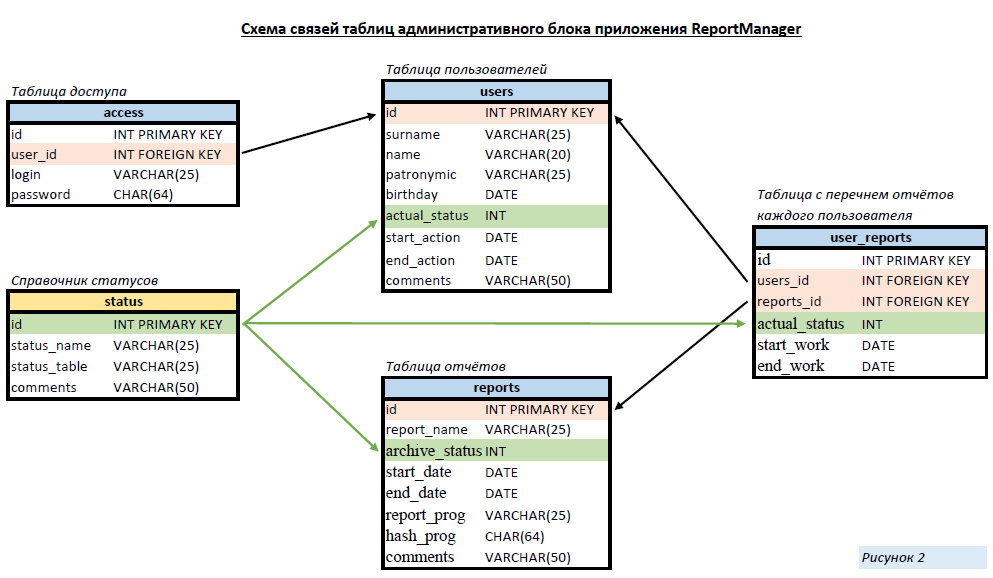
* уникальный первичный ключ (id);
* внешний ключ, связь с таблицей users (users\_id);
* внешний ключ, связь с таблицей reports (reports\_id);
* статус актуальности – работает ли еще данный пользователь с этим отчетом или уже нет, если работает, то поле содержит единицу, если отчёт передали другому исполнителю, то поле содержит другое значение, которое определено в справочнике статусов (status);
* дата начала работы пользователя с отчётом (start\_work);
* дата окончания работы пользователя с отчётом (end\_work);

Поскольку для таблиц «users», «reports», «user\_reports» отслеживается статус актуальности записей (поле «status»), в административном блоке ReportManager введён справочник статусов – таблица-справочник «status», которая содержит три поля:

* уникальный первичный ключ (id);
* название статуса (status\_name);
* поле для комментариев (comment);

Схема работы административного блока приложения ReportManager представлена ERD-диаграммой на рисунке 2.

Рисунок 2



Второй блок приложения ReportManager посвящен коммерческой и хозяйственной деятельности банка (предприятия). Поскольку воспроизвести реальную рабочую систему учета банка проблематично по понятным причинам, здесь, в дипломном проекте, для демонстрационных целей, работа приложения ReportManager будет показана на упрощенном наборе данных. Места упрощения будут показаны дальше. Однако, несмотря на упрощение, набор таблиц для учета коммерческой деятельности банка будет практически «боевой». Итак, для хранения основной информации банка по его коммерческой деятельности требуется три основные таблицы: таблица с планом счетов банковского бухгалтерского учёта – «accounts», таблица учёта остатков «saldo» и таблица для хранения транзакций «transacts».

Таблица «accounts» является основой для учёта коммерческой деятельности любого предприятия, в том числе и банка, поскольку содержит информацию о плане счетов бухгалтерского учёта. На основании этого плана и осуществляется учёт коммерческой деятельности банка. Таблица «accounts» хранит информацию в следующих полях:

* номер счета, уникальный первичный ключ (id);
* название счёта (name);
* группа счёта (group\_accounts);
* признак активности/пассивности счёта (ap\_status);
* статус счета (status);
* дата начала работы счёта (start\_date);
* дата окончания работы счёта (end\_date);

Основной таблицей для создания отчётности банка является таблица «saldo», а точнее её содержимое, которое представляет собой остаток средств на каждом счёте на конец каждого рабочего дня. Информация об остатках средств хранится в 4-х полях:

* первичный ключ (id);
* номер счёта (account);
* дата, на конец дня которой зафиксирован остаток по счёту (sdate);
* остаток по счёту в рублях (summa);

Таблица «saldo» хранит остатки на всех счетах в рублях, в том числе и на валютных счетах. В реальной рабочей схеме должна быть реализована таблица курсов, хранящая значения официального курса валюты от ЦБ на каждый рабочий день, с помощью которой можно получить остатки на валютных счетах в каждой конкретной валюте на любой определенный день – в настоящем проекте эта таблица отсутствует, это одно из упрощений о которых говорилось выше.

Третья таблица является главной с точки зрения хранения информации по учёту коммерческой деятельности банка – она содержит всю информацию по транзакциям (или, иначе, бухгалтерским проводкам банка), которые и являются отражением совершённых банком операций. Для построения отчётности таблица «transacts» используется реже, чем таблица остатков «saldo», но, тем не менее, используется. Таблица состоит из 7 полей:

* уникальный первичный ключ (id);
* дата и время транзакции (tdate);
* счёт по дебету транзакции (accdebet);
* счёт по кредиту транзакции (acccredit);
* сумма транзакции (summa);
* номер первичного документа (docid);
* комментарий (comment);

Еще одно упрощение реального набора данных, о котором говорилось выше, заключается в том, что в данной упрощенной модели отсутствует таблица с первичными документами учёта на основании которых и осуществляется та или иная транзакция (поэтому поле «docid» из таблицы «transacts», которое должно было бы в реальной рабочей модели отвечать за связь с таблицей первичных документов, здесь имеет значение символического атрибута). А также, как и для таблицы «saldo», валюта транзакций – рубль, валютные транзакции должны были бы вычисляться исходя из данных таблицы курсов. Кроме того, сокращен перечень счетов бухгалтерского учета в таблице «accounts», однако присутствующие в таблице счета обеспечивают поддержание учетных данных в состоянии баланса и позволяют продемонстрировать примеры работы с отчетностью в приложении ReportManager.

**Глава** 3.

3. Работа с программой.

3.1. Основной модуль, вход в программу.

При запуске программа отображает основную форму и, поверх основной, форму для идентификации пользователя, Основная форма при этом не активна (рисунок 3.1).

После ввода логина и пароля, нажимая кнопку «Войти», пользователь запускает проверку пароля, при этом пароль введённый пользователем преобразуется в hash-число и сравнивается с hash-числом пароля этого пользователя, хранящимся в базе данных в таблице «access». Соответствующий hash находим с помощью логина, хранящегося в этой же таблице. Если в результате сравнения пароли совпали, то форма для идентификации закрывается, активной становится основная форма и пользователь получает доступ к перечню своих отчётов (рис. 3.2)

Рисунок 3.1

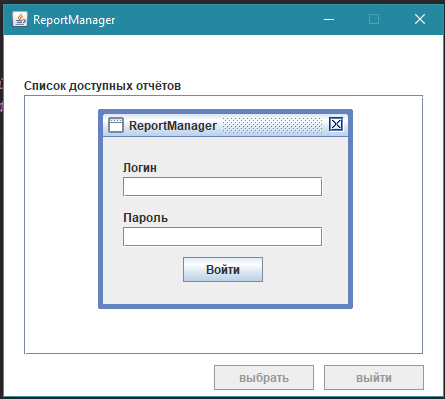
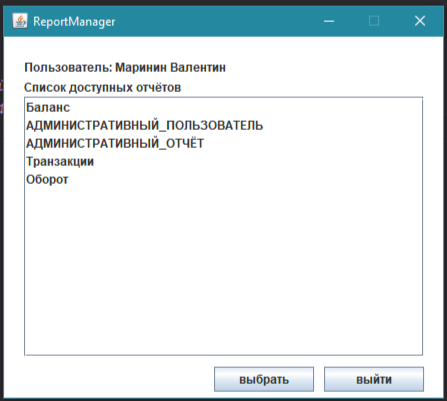


Рисунок 3.2



Перечень отчётов пользователя получаем из базы данных, используя связку таблиц «access», «user\_reports», «reports». Связку можно отследить на рисунке 2.

Далее пользователь двойным щелчком мыши на нужном отчёте или кнопкой «выбрать» вызывает подпрограмму, которая предназначена для получения выбранного отчёта. Эта подпрограмма по введённым пользователем исходным данным (дата отчета или период, номер счёта и т.п.) отправит запрос в базу данных, а затем из полученных данных сформирует отчёт и сохранит его в нужном формате. Это если речь идёт о группе отчётов, связанных с профильной деятельностью банка или предприятия. Если же это административная группа подпрограмм, связанных с управлением пользователями и отчетами, то тогда результатом работы подпрограммы будет добавление нового пользователя, редакция данных уже существующего пользователя, добавление в базу данных сведений о новом отчёте и так далее. Более подробно это описано в разделе 1.2.

Для работы пользователю одномоментно разрешено вызывать на исполнение и работать только с одним отчётом, такое положение реализовано исходя из моего опыта работы - пользователи склонны запускать одновременно все имеющиеся в их распоряжении отчёты на получение данных ради экономии времени, что приводило не только к «зависанию» локальной машины, но и перегружало базу данных. А это в свою очередь осложняло работу фронт-офиса, замедляло работу с клиентами, что недопустимо. При желании, конечно, можно легко модифицировать основной модуль, чтобы запускать несколько отчётов одномоментно.

При вызове отчёта (двойной клик или кнопка «выбрать») окно основной программы ReportManager со списком отчётов получает статус Visible(false) и ожидает окончания работы вызванного отчёта, после чего снова получает статус Visible(true) до вызова следующего отчёта или завершения работы программы.

3.2. Группа административных подпрограмм-отчётов.

Таких подпрограмм две, на рисунке 3.2 их названия выведены в верхнем регистре:

* АДМИНИСТРАТИВНЫЙ\_ПОЛЬЗОВАТЕЛЬ
* АДМИНИСТРАТИВНЫЙ\_ОТЧЁТ

3.2.1. АДМИНИСТРАТИВНЫЙ\_ПОЛЬЗОВАТЕЛЬ

Интерфейс

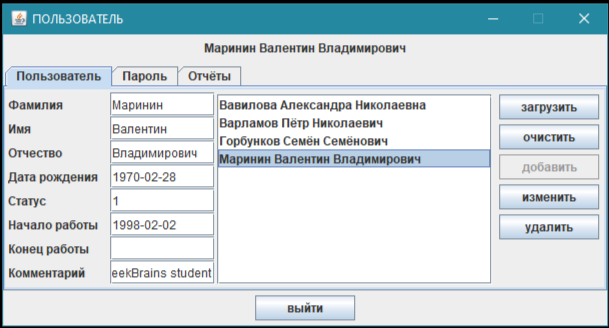
Подпрограмма reportUser.jar (в ReportManager имеет название «АДМИНИСТРАТИВНЫЙ\_ПОЛЬЗОВАТЕЛЬ») отвечает за добавление нового пользователя в базу данных, а также за редактирование и удаление пользователей. Эта же подпрограмма добавляет и редактирует пароли пользователей, а кроме того управляет набором отчётов доступных каждому пользователю.

Интерфейс подпрограммы построен исходя из этих трёх составляющих подпрограммы – основная форма имеет три закладки: «Пользователь», «Пароль», «Отчёты». Внешний вид представлен на рисунке 3.3 – с активной закладкой «Пользователь». Эта подпрограмма при работе с базой данных использует библиотеку hibernate, поэтому каждая из трёх закладок (точнее говоря класс, данные которого отражены на закладке) в программном плане представляет собой отдельную сущность, связанную со своей таблицей в базе данных. Для закладки «Пользователь» используется таблица «users» базы данных «enterprise», для закладки «Пароль» - таблица «access» и для закладки «Отчёты» - «user\_reports» и «reports».

*Закладка «Пользователь»*

Работа с подпрограммой начинается с нажатия кнопки «Загрузить», при этом из базы данных загружается список всех пользователей предприятия. Список, как видно на изображении, расположен в центре. Щелчок мышкой на каком-нибудь конкретном пользователе выделяет строку в списке и выводит детальную информацию по выделенному пользователю в панель расположенную слева от списка. При изменении выделенной строки в списке (мышкой или клавишами стрелками) будет изменяться и детализированная информация о пользователе в левой панели. При выделении любого пользователя деактивируется кнопка «добавить», чтобы не происходило повторное добавление существующего пользователя в базу данных. Кроме того при изменении выделенного пользователя в списке изменяется текущий активный пользователь, который выводится в верхней панели подпрограммы над списком пользователей. Текущий активный пользователь виден при активации любой закладки, он используется для ввода или редактирования пароля, а также при назначении или редактировании его списка отчётов.

Кнопка «очистить» очищает левую панель от детализированных данных выделенного в списке пользователя, снимает выделение с этого пользователя в списке пользователей и устанавливает значение текущего активного пользователя в null – верхняя панель при этом становится пустой. Собственно, точно такое же состояние возникает сразу после загрузки этой административной подпрограммы. Такое состояние необходимо для ввода нового пользователя в базу данных. Рисунок 3.3

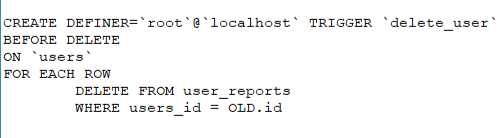


Для того, чтобы ввод состоялся, необходимо заполнить поля с фамилией, именем и отчеством нового пользователя, датой его рождения и датой начала его трудовой деятельности на данном предприятии. Статус выставляется программой, значение «1» означает, что пользователь активен, а значит должен иметь пароль и ему может быть присвоен некий набор отчётов для выполнения. Кроме того можно заполнить поле комментария, но это необязательное поле для введения нового пользователя в базу данных. Также нажатие кнопки «очистить» активирует кнопку «добавить» и деактивирует кнопки «изменить», «удалить».

Кнопка «добавить» добавляет нового пользователя в базу данных. При добавлении пользователя в таблицу «users» базы «enterprise» добавляется новая запись. Для успешного добавления необходимо заполнить ФИО и даты рождения и начала работы на предприятии, без этих данных ввод нового пользователя невозможен. Даты проверяются на корректность, если формат ввода («ГГГГ-ММ-ДД») некорректен или присутствуют буквы в дате – ввод также не состоится.

Кнопка «изменить» вносит изменения по текущему активному пользователю (отражается на верхней панели подпрограммы), для изменений доступны все поля на панели детальной информации. При нажатии кнопки выполняется проверка заполнение обязательных полей: фамилии, имени, отчества, даты рождения и даты начала работы на предприятии. Одновременно проверяется, что список пользователей не пустой – иначе редактировать некого и если заполнить данные в панели детальной информации, но при этом будет пустой список, то кнопка редактирования не сработает. Такая ситуация характерна для ввода нового пользователя и поэтому в этой ситуации должна работать кнопка «добавить».

Кнопка «удалить» удаляет пользователя из таблицы «users». Удаление пользователя влечет за собой удаление его данных не только из таблицы «users», но и из связанных с пользователем таблиц «access» (логин и пароль) и «user\_reports» (набор отчётов пользователя). Удаление данных в первую очередь происходит из таблицы «access», это удаление выполняется средствами Java, используя возможности библиотеки hibernate, удаляется не только запись из «access», но и удаляется аккаунт пользователя из базы данных, используя директиву «DROP USER <USER\_LOGIN> ». Далее перед удалением пользователя из таблицы «users» удаляется набор его отчётов из таблицы «user\_reports», реализовано это удаление с помощью средств базы данных – на таблице «users» реализован триггер типа BEFORE DELETE, который срабатывает при попытке удаления пользователя из таблицы «users», в тело триггера(хранимой процедуры) передаётся id удаляемого пользователя по которому и определяется набор отчётов удаляемого пользователя.



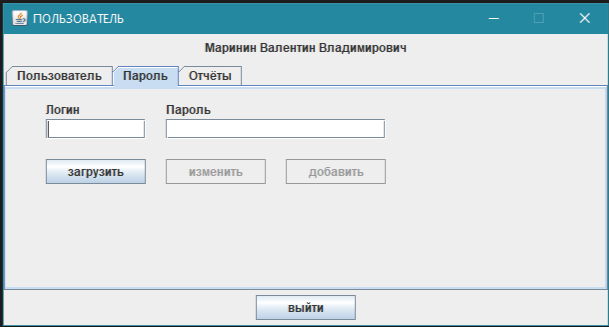
*Закладка «Пароль»*

После создания нового пользователя на закладке «Пользователь», необходимо создать аккаунт нового пользователя в базе данных, создать его логин и пароль, для этого переключаемся на закладку «Пароль» (рисунок 3.4).

Кнопка «загрузить». После ввода нового пользователя или после изменения текущего активного пользователя закладка «Пароль» приводится к состоянию, которое отражено на рисунке 3.4. Активна только кнопка «загрузить». По нажатию кнопки осуществляется поиск текущего активного пользователя (отражается на верхней панели) в таблице «access» по id пользователя из таблицы «users». Если запись с искомым id существует в «access», то тогда в соответствующие окна на закладке выводятся логин и hash-число пароля. При этом активируются кнопки «добавить» и «изменить». Если же запись с таким id не найдена – то есть это новый пользователь, выводится сообщение о том, что пароль и логин для данного пользователя не определены, при этом активируется только одна кнопка «добавить».

Кнопка «добавить». По нажатию этой кнопки происходит добавление аккаунта пользователя в базу данных с указанными логином и паролем – используется директива «CREATE USER <user\_LOGIN> IDENTIFIED BY <PASSWORD>». Сразу же пользователю выдаются права на выборку, выполнение и создание директивой «GRANT SELECT, EXECUTE, CREATE ON enterprise.\* TO <USER\_LOGIN>».

Рисунок 3.4



Далее этот логин и пароль вносятся записью в таблицу «access», пароль перед записью в таблицу шифруется с помощью hash функции SHA-256 (Secure Hash Algorithm Version 2 (англ.) – Алгоритм Безопасного Хэширования).

Хеш-функция [3] ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) hash function от hash — «превращать в фарш», «мешанина»), или функция свёртки — функция, осуществляющая преобразование [массива](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) входных данных произвольной длины в выходную [битовую](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82) строку установленной длины, выполняемое [определённым алгоритмом](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC). Преобразование, производимое хеш-функцией, называется хешированием. Исходные данные называются входным массивом, «ключом» или «сообщением». Результат преобразования называется «[хешем](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%81%D1%83%D0%BC%D0%BC%D0%B0)», «хеш-кодом», «хеш-суммой», «сводкой [сообщения](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B5)».

Эти логин и пароль пользователь вводит в окно идентификации программы ReportManager (см. рисунок 3.1). Программа шифрует введённый пользователем при запуске программы пароль и сравнивает полученный хэш-код с хэш-кодом хранящимся в таблице «access», предварительно найдя его по введённому логину. Если логин не найден или хэш-код введённого пароля не совпадает с табличным – вход не состоится, программа выдаст сообщение об ошибке.

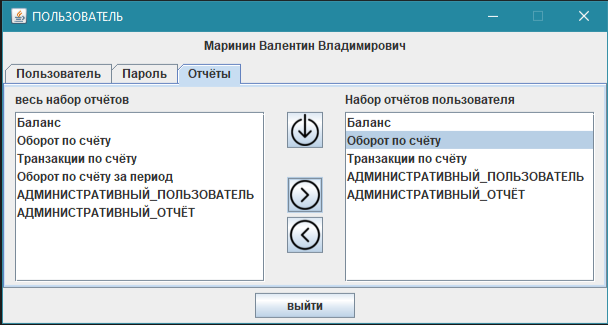
Если пользователь забыл пароль, то нужно будет воспользоваться кнопкой «изменить» на закладке «Пароль» и ввести там новый пароль. Однако фактически пароль будет обновлён только в таблице «access», а для замены пароля для аккаунта пользователя в базе данных произойдёт удаление существующего аккаунта пользователя с логином текущего пользователя и будет создан новый аккаунт с таким же логином и вновь введённым паролем.

После успешного входа в программу (см. рисунок 3.2) логин и пароль будут передаваться параметрами в каждый отчёт, который будет вызван пользователем на исполнение. В вызванном отчёте логин и пароль будут использованы при подключении к базе данных.

*Закладка «Отчёты»*

На этой закладке происходит управление пользовательским набором отчётов – то есть теми отчётами, которые пользователь может вызвать на исполнение, весь свой набор пользователь видит сразу после входа в программу (см. рисунок 3.2).

Рисунок 3.5



На закладке два набора отчётов – полный набор отчётов существующих в программе, этот набор хранится в таблице «reports» и набор отчётов пользователя, этот набор хранится в таблице «user\_reports».

На закладке три кнопки. Верхняя кнопка служит для загрузки обоих наборов, если пользователь новый, то его набор (правое окно) будет пустым. Две нижние кнопки добавляют и убирают выбранный отчёт из одного списка в другой.

При смене активного текущего пользователя (выводится на верхней панели над закладками) оба списка обнуляются, точно также как и сбрасывается вся информация на закладке «Пароль». То есть при смене активного пользователя нужно каждый раз отдельно загружать информацию и для закладки «Пароль», и для закладки «Отчёты».

Реализация

UML-диаграмма классов приведена в приложении №1 страница 52 (UML ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Unified Modeling Language - унифицированный язык моделирования) — язык [графического](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) описания для [объектного моделирования](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) в области [разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), для [моделирования бизнес-процессов](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D0%B2), [системного проектирования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и отображения [организационных структур](https://ru.wikipedia.org/wiki/%D0%9E%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0)).

Подпрограмма-отчёт организационно разбита на четыре пакета: «model», «view», «presenter» и «repository».

**Пакет «model»** содержит четыре класса-сущности связанные с таблицами базы данных «enterprise»:

* класс User – таблица «users», данные отображаются на закладке «Пользователь»;
* класс «UserLoginData» - таблица «access», работа с данными происходит на закладке «Пароль»;
* класс «FullReportsSet» - таблица «reports», предоставляет полный набор отчётов для закладки «Отчёты»:
* класс «UserReportsSet» - таблица «user\_reports», набор отчётов пользователя, также для закладки «Отчёты»;

Также пакет «model» содержит класс «HashCount», этот класс рассчитывает hash-код для пароля пользователя, а также рассчитывает hash-код для подпрограмм отчётов, чтобы обеспечить версионность этих подпрограмм.

**Пакет «view»** согласно названию призван отображать данные введённые пользователем или полученные из презентера и передавать введённые данные в презентер для дальнейшей обработки. Для каждой закладки используется свой класс, базовым классом, который собирает всё воедино является «ViewUserReport».

* классы «DetailUserPanel» и «ListUserPanel» - используются для отображения данных закладки «Пользователь»;
* класс «LoginDataPanel» - закладка «Пароль»;
* класс «UserReportsPanel» - списки отчётов на закладке «Отчёты»;

В пакете «view» для обеспечения архитектурной границы введены три интерфейса (по числу закладок):

* «iViewDetailsData» - этот интерфейс имплементирует класс «ViewUserReport», интерфейс содержит четыре метода: load, add, edit, delete – эти методы передают в презентер информацию введённую пользователем или размещают информацию полученную от презентера в форме на закладке «Пользователь», а также выполняют предварительные проверки на полноту передаваемой информации;
* «iViewLoginData» - этот интерфейс имплементируется классом «loginDataPanel», содержит такие же четыре метода метода: load, add, edit, delete. Методы обрабатывают информацию закладки «Пароль»;
* «iViewSetReports» - данный интерфейс имплементирован классом «UserReportsPanel», декларирует методы: getWholeSetReports, getUserSetReports, addReportToUser, removeReportFromUser. Эти методы работают с информацией на закладке «Отчёты»;

Архитектурные границы необходимы для нескольких целей:

- организационные (каждый должен иметь свою зону ответственности, определенную границей);

- черный ящик (чем меньше деталей мы знаем, тем лучше);

- отделение значимого, от незначимого (в один момент времени мы можем сосредоточиться только на чем-то одном);

- разделение постоянных частей и временных (постоянные - ядро, бизнес-логика, GUI - временные решения);

Границы нужны чтобы их пересекать, для этого вводятся правила (интерфейсы) взаимодействия. На уровне кода - это способность через интерфейс обратиться к функциям за пределами границы.

**Пакет «repository»** традиционно отвечает за связь с базой данных. Содержит четыре интерфейса для обеспечения архитектурной границы и четыре класса реализующие эти интерфейсы. Интерфейсы:

* «iUserDetailRepo» содержит методы: add, delete, update, getAll, getById, closeDB. Интерфейс имплементируется классом «UserDetailDB» и работает с информацией закладки «Пользователь», которая в базе данных хранится в таблице «users»;
* «iLoginDataRepo» декларирует методы: getById, addById, updateById, deleteById, addDBuser, deleteDBuser, closeDB. Имплементирует данный интерфейс класс «UserLoginDB», особенность работы данного класса в том, что он работает не только с рабочей базой проекта «enterprise», но и с системными таблицами во время создания и удаления пользовательских аккаунтов, а также выделения прав доступа новым пользователям. При этом в базе данных «enterprise» задействована таблица «access», а изменения в системных данных затрагивают, в частности, таблицу «mysql.user». вся работа с данными инициируется с закладки «Пароль»;
* «iFullSetReportsRepo» имеет два метода: getFullSetReports и closeDB/ Имплементируется классом «FullSetReportsDB», который предоставляет данные о полном наборе отчётов организации для закладки «Отчёты», информация извлекается из таблицы «enterprise.reports»;
* «iUserSetReportsRepo» декларирует четыре метода: getUserReports, addReport, deleteReport, closeDB. Имплементируется классом «UserSetReportsDB», который управляет пользовательским набором отчётов, оперируя данными таблицы «user\_reports». Информация также размещается на закладке «Отчёты»;

Пакет «presenter» как и все остальные пакеты содержит интерфейсы для реализации архитектурных границ. Для использования методов из классов-презентеров в трех классах пакета «view» объявляется поле с типом соответствующего интерфейса-презентера. После создания класса из пакета «view» методом SetPresenter инициализируется поле имеющее тип интерфейса-презентера, инициализацию этого поля невозможно сделать в конструкторе, так как для создания презентера требуется уже созданный класс из пакета «view». Например, в классе «ViewUserReport» объявлено поле с типом «iUserDetailsPresenter» в момент инициализации этого поля через метод setPresenter() в этот метод параметром передаётся ссылка на класс из пакета «view» (в данном случае это класс «ViewUserReport», передаём как «this») для которого и создаётся этот презентер, строка реализации выглядит следующим образом:

*setPresenter( new UserDetailsPresenter(this, new UserDetailDB("userDetail.cfg.xml")) );*

Методы презентера вызываются в обработчиках нажатия оконных кнопок.

Интерфейсов в пакете три (по числу закладок):

* «iUserDetailsPresenter» содержит методы: load, add, delete, update, closeDB. Этот интерфейс имплементирован классом «UserDetailPresenter». Этот класс-презентер обеспечивает обработку запросов поступающих из пакета «view» от класса «ViewUserReport», а для получения данных из базы или сохранения данных подключается класс «UserdetailDB»;
* «iUserLoginPresenter» декларирует методы: loadUserLoginData, addUserLoginData, updateUserLoginData, deleteUserLoginData, closeDB. Этот интерфейс реализован в классе «UserLoginPresenter». К этому презентеру подключаются классы: «LogindataPanel» пакет «view» и «UserLoginDB» пакет «repository».
* «iSetReportsPresenter» имеет методы: loadUserSetReports, loadFullSetReports, addReportToUser, removeReportFromUser, closeDB. В классе «SetReportsPresenter», имплементирующем данный интерфейс, к презентеру подключаются три класса: «UserReportsPanel» пакет «view» и классы «FullSetReportsDB», «UserSetReportsDB» из пакета «repository».

3.2.2. АДМИНИСТРАТИВНЫЙ\_ОТЧЁТ

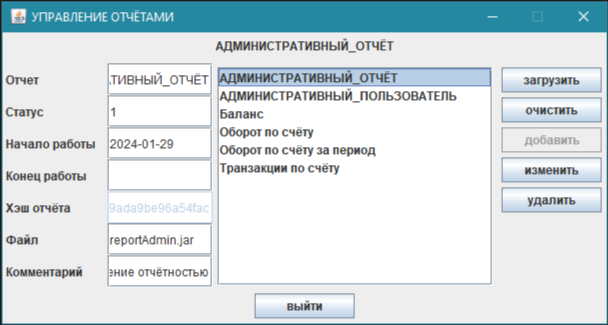
Интерфейс

Подпрограмма ***reportAdmin.jar*** (в ReportManager имеет название АДМИНИСТРАТИВНЫЙ\_ОТЧЁТ) отвечает за добавление нового отчёта в базу данных, а также за редактирование и удаление отчётов, кроме того для новых и модифицированных отчётов пересчитывается hash-код, что необходимо для работы механизма контроля версий – при старте программы ReportManager основной модуль ***ReportManager.jar*** сравнивает hash-код подпрограмм-отчётов хранящихся на сетевом ресурсе и на локальной машине и при несовпадении этих hash-кодов происходит копирование сравниваемого файла с сетевого ресурса на локальную машину, копирование происходит и при отсутствии файла подпрограммы отчёта на локальной машине (эта ситуация возникает когда вводится новый отчёт).

Интерфейс подпрограммы схож с интерфейсом подпрограммы-отчёта АДМИНИСТРАТИВНЫЙ\_ПОЛЬЗОВАТЕЛЬ, внешний вид подпрограммы приведён на рисунке 4.1. Работа с подпрограммой также начинается с нажатия кнопки «Загрузить», при этом из базы данных «enterprise» таблица «reports» загружается список всех отчётов предприятия, реализованных в приложении ReportManager. Список отчётов расположен в центре формы. Щелчок мышкой на каком-нибудь конкретном отчёте выделяет строку в списке и выводит детальную информацию по выделенному отчёту в панель расположенную слева от списка. При изменении выделенной строки в списке (мышкой или клавишами стрелками) будет изменяться и детализированная информация об отчёте в левой панели.

При выделении строки с отчётом деактивируется кнопка «добавить», чтобы не происходило повторное добавление в базу данных отчёта, который уже существует в базе данных. Кроме того, при изменении выделенного отчёта в списке изменяется текущий активный отчёт, который выводится в верхней панели подпрограммы над списком отчётов. Текущий активный отчёт используется для редактирования данных и для пересчёта hash-кода файла содержащего подпрограмму-отчёт, когда такая подпрограмма была модифицирована и её нужно распространить на машины пользователей.

Рисунок 4.1

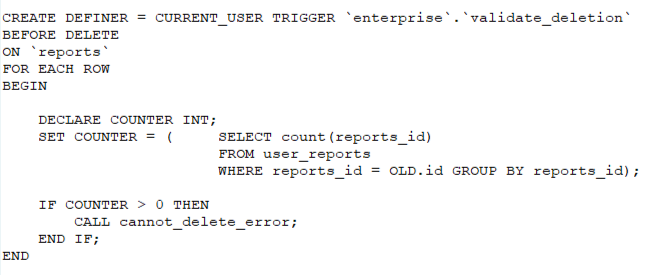


Кнопка «очистить» очищает левую панель от детализированных данных по отчёту выделенному в списке, снимает выделение с отчёта в списке отчётов (находящемуся в центре формы) и устанавливает значение активного отчёта в null – верхняя панель при этом становится пустой. Такое состояние возникает сразу после загрузки этой административной подпрограммы и оно необходимо для ввода нового отчёта в базу данных.

Кнопка «добавить». По нажатию этой кнопки происходит добавление данных вновь введённого отчёта в базу данных «enterprise» в таблицу «reports». Для успешного добавления необходимо заполнить поля с названием отчёта, датой начала работы отчёта и названием файла, который является подпрограммой-отчётом реализующей данный отчёт, без этих данных ввод нового отчёта не состоится. Дата начала работы отчёта проверяется на корректность, если формат ввода («ГГГГ-ММ-ДД») некорректен или присутствуют буквы в дате – ввод также не состоится.

Кнопка «изменить» вносит изменения по текущему активному отчёту (отражается на верхней панели подпрограммы), для изменений доступны все поля на панели детальной информации. При нажатии кнопки выполняется проверка на заполнение обязательных полей: название отчёта (выводится в списке отчётов), имени файла содержащего подпрограмму-отчёт и даты начала работы отчёта в рамках ReportManager. Одновременно проверяется, что список пользователей не пустой – иначе редактировать нечего и если заполнить данные в панели детальной информации, но при этом будет пустой список, то кнопка редактирования не сработает. Ситуация при которой поля панели с детальной информацией заполнены, а список отчётов в этот момент пуст, характерна для ввода нового отчёта и поэтому в этой ситуации должна работать кнопка «добавить».

Кнопка «удалить» удаляет выбранный отчёт из таблицы «reports». Удаление отчёта влечет за собой удаление его данных из таблицы «reports». Однако предварительно выполняется проверка на наличие ссылок на этот отчёт в таблице связанной с пользователем - «user\_reports» (набор отчётов пользователя). Если ссылки обнаруживаются, то это означает, что этот отчёт используется каким-то пользователем и не может быть удалён до тех пор, пока его (отчёт) использует хотя бы один пользователь. Если же ссылок не обнаружено – тогда соответствующая запись удаляется из списка отчетов (таблицы «reports»). Проверка на наличие ссылок выполняется средствами MySQL – реализован триггер «validate\_deletion».



При наличии записей в «user\_reports» c идентификатором удаляемого отчёта генерируется ошибка «cannot\_delete\_error» и удаление отменяется, пользователю выводится сообщение о том, что отчёт используется и не может быть удалён. Если же записи в «user\_reports» не обнаружены, то отчёт удаляется из приложения ReportManager.

Кнопка «персонал» выводит список пользователей использующих выделенный отчёт. Этот список необходим, если нужно удалить отчёт, но удаление не происходит, поскольку отчёт присутствует в таблице «user\_reports», что означает, что он используется каким-то пользователем. По нажатия кнопки «персонал» находим пользователей использующих данный отчёт и с помощью подпрограммы «АДМИНИСТРАТИВНЫЙ\_ПОЛЬЗОВАТЕЛЬ» убираем отчёт подлежащий удалению из списка отчётов пользователя. После того как удаляемый отчёт будет убран из персональных списков всех пользователей (то есть в таблице «user\_reports» не останется записей с идентификатором удаляемого отчёта), отчет можно будет удалить.

Реализация

Реализация подпрограммы «АДМИНИСТРАТИВНЫЙ\_ОТЧЁТ» схожа с реализацией подпрограммы «АДМИНИСТРАТИВНЫЙ\_ПОЛЬЗОВАТЕЛЬ». Отличие заключается в том, что в данной подпрограмме не использовались закладки, соответственно уменьшилось количество кода.

Подпрограмма-отчёт организационно разбита на четыре пакета: «model», «view», «presenter» и «repository».

**Пакет «model»** содержит три класса-сущности связанные с таблицами базы данных «enterprise»:

* класс «Report» - таблица «reports», предоставляет полный набор отчётов организации, автоматизированный в рамках приложения ReportManager;
* класс User – таблица «users», данные отображаются по нажатию кнопки «персонал»;
* класс «UserReportsSet» - таблица «user\_reports», набор отчётов пользователя, необходимо для работы кнопки «персонал»;

Также пакет «model» содержит класс «HashCount», этот класс рассчитывает hash-код для подпрограмм отчётов, чтобы обеспечить версионность этих подпрограмм.

**Пакет «view»** это оконная форма для отображения данных введённых пользователем или полученных из презентера, а также для передачи введённых пользователем данных (новый отчёт или корректировка существующего) в презентер для дальнейшей обработки. В оконной форме используется интерфейс схожий с интерфейсом отчета «АДМИНИСТРАТИВНЫЙ\_ПОЛЬЗОВАТЕЛЬ» закладка «Пользователь». Интерфейс образован рядом панелей (см. рисунок 4.1):

* верхняя панель отображает текущий активный отчёт;
* нижняя панель содержит кнопку выхода из подпрограммы;

Центральная базовая панель содержит три панели образующих всю содержательную часть подпрограммы:

* в центре базовой панели расположена панель со списком всех отчётов реализованных в рамках приложения ReportManager, отчёт выделенный в списке является текущим активным отчётом, его название отображается в верхней панели;
* слева расположена панель с детальной информацией по текущему выделенному отчёту, таким образом, изменяя информацию в левой панели можно корректировать информацию по текущему отчёту (сохранение скорректированной информации происходит по нажатию кнопки «изменить»). Кроме того, эта же панель используется для ввода информации по новому отчёту (см. описание по кнопке «добавить»);
* справа расположена панель с управляющими списком отчётов кнопками кнопками

В пакете «view» для обеспечения архитектурной границы введен три интерфейса (по числу закладок):

* «iViewDetailsData» - этот интерфейс имплементирует класс «ViewUserReport», интерфейс содержит четыре метода: load, add, edit, delete – эти методы передают в презентер информацию введённую пользователем или размещают информацию полученную от презентера в форме на закладке «Пользователь», а также выполняют предварительные проверки на полноту передаваемой информации;
* «iViewLoginData» - этот интерфейс имплементируется классом «loginDataPanel», содержит такие же четыре метода метода: load, add, edit, delete. Методы обрабатывают информацию закладки «Пароль»;
* «iViewSetReports» - данный интерфейс имплементирован классом «UserReportsPanel», декларирует методы: getWholeSetReports, getUserSetReports, addReportToUser, removeReportFromUser. Эти методы работают с информацией на закладке «Отчёты»;

Архитектурные границы необходимы для нескольких целей:

- организационные (каждый должен иметь свою зону ответственности, определенную границей);

- черный ящик (чем меньше деталей мы знаем, тем лучше);

- отделение значимого, от незначимого (в один момент времени мы можем сосредоточиться только на чем-то одном);

- разделение постоянных частей и временных (постоянные - ядро, бизнес-логика, GUI - временные решения);

Границы нужны чтобы их пересекать, для этого вводятся правила (интерфейсы) взаимодействия. На уровне кода - это способность через интерфейс обратиться к функциям за пределами границы.

**Пакет «repository»** традиционно отвечает за связь с базой данных. Содержит четыре интерфейса для обеспечения архитектурной границы и четыре класса реализующие эти интерфейсы. Интерфейсы:

* «iUserDetailRepo» содержит методы: add, delete, update, getAll, getById, closeDB. Интерфейс имплементируется классом «UserDetailDB» и работает с информацией закладки «Пользователь», которая в базе данных хранится в таблице «users»;
* «iLoginDataRepo» декларирует методы: getById, addById, updateById, deleteById, addDBuser, deleteDBuser, closeDB. Имплементирует данный интерфейс класс «UserLoginDB», особенность работы данного класса в том, что он работает не только с рабочей базой проекта «enterprise», но и с системными таблицами во время создания и удаления пользовательских аккаунтов, а также выделения прав доступа новым пользователям. При этом в базе данных «enterprise» задействована таблица «access», а изменения в системных данных затрагивают, в частности, таблицу «mysql.user». вся работа с данными инициируется с закладки «Пароль»;
* «iFullSetReportsRepo» имеет два метода: getFullSetReports и closeDB/ Имплементируется классом «FullSetReportsDB», который предоставляет данные о полном наборе отчётов организации для закладки «Отчёты», информация извлекается из таблицы «enterprise.reports»;
* «iUserSetReportsRepo» декларирует четыре метода: getUserReports, addReport, deleteReport, closeDB. Имплементируется классом «UserSetReportsDB», который управляет пользовательским набором отчётов, оперируя данными таблицы «user\_reports». Информация также размещается на закладке «Отчёты»;

**Пакет «presenter»** как и все остальные пакеты содержит интерфейсы для реализации архитектурных границ. Для использования методов из классов-презентеров в трех классах пакета «view» объявляется поле с типом соответствующего интерфейса-презентера. После создания класса из пакета «view» методом SetPresenter инициализируется поле имеющее тип интерфейса-презентера, инициализацию этого поля невозможно сделать в конструкторе, так как для создания презентера требуется уже созданный класс из пакета «view». Например, в классе «ViewUserReport» объявлено поле с типом «iUserDetailsPresenter» в момент инициализации этого поля через метод setPresenter() в этот метод параметром передаётся ссылка на класс из пакета «view» (в данном случае это класс «ViewUserReport», передаём как «this») для которого и создаётся этот презентер, строка реализации выглядит следующим образом:

*setPresenter( new UserDetailsPresenter(this, new UserDetailDB("userDetail.cfg.xml")) );*

Методы презентера вызываются в обработчиках нажатия оконных кнопок.

Интерфейсов в пакете три (по числу закладок):

* «iUserDetailsPresenter» содержит методы: load, add, delete, update, closeDB. Этот интерфейс имплементирован классом «UserDetailPresenter». Этот класс-презентер обеспечивает обработку запросов поступающих из пакета «view» от класса «ViewUserReport», а для получения данных из базы или сохранения данных подключается класс «UserdetailDB»;
* «iUserLoginPresenter» декларирует методы: loadUserLoginData, addUserLoginData, updateUserLoginData, deleteUserLoginData, closeDB. Этот интерфейс реализован в классе «UserLoginPresenter». К этому презентеру подключаются классы: «LogindataPanel» пакет «view» и «UserLoginDB» пакет «repository».
* «iSetReportsPresenter» имеет методы: loadUserSetReports, loadFullSetReports, addReportToUser, removeReportFromUser, closeDB. В классе «SetReportsPresenter», имплементирующем данный интерфейс, к презентеру подключаются три класса: «UserReportsPanel» пакет «view» и классы «FullSetReportsDB», «UserSetReportsDB» из пакета «repository».

3.3. Группа управленческих, бухгалтерских и статистических подпрограмм-отчётов.

Все отчёты, входящие в эту группу, отражают разнообразные показатели коммерческой деятельности организации и призваны выполнять контрольную, надзорную, информационную функции как для руководства организации, так и для контролирующих органов – так Центробанк выполняет надзорную и контрольную функции по отношению к коммерческим банкам и с помощью обязательной банковской отчётности проверяет соблюдение банками так называемых пруденциальных нормативов.

Реализация каждого отдельно взятого отчёта в отдельной подпрограмме, скомпилированной в итоге в исполняемый jar-файл, позволяет легко и достаточно просто модифицировать каждый отчёт. Разработка же нового отчёта, учитывая, что подпрограмма, реализующая отчёт и далее компилируемая в jar-файл, имеет достаточно однотипную структуру, занимает минимальное количество времени, так как за основу можно взять какой-то уже имеющийся отчёт.

Группа управленческих, бухгалтерских и статистических отчётов может быть условно разделена на две подгруппы (это отражено на рисунке 1):

* отчёты с произвольной структурой (в основном управленческие отчёты);
* отчёты с заранее предопределённой структурой (подавляющее большинство бухгалтерских и статистических отчётов);

3.3.1. Отчёты с произвольной структурой.

Отчёты с произвольной структурой проще в реализации, так как вся необходимая для отчёта информация чаще всего может быть получена простой операцией выборки (SQL-select), а также достаточно простой, по сравнению со второй группой (группа отчётов с предопределённой структурой), вывод полученной информации в желаемой форме. Обычно такие отчёты являются управленческими, то есть внутренними отчётами. Эти отчёты предоставляются руководству организации (менеджеры разных уровней) для оперативного контроля за текущим состоянием организации.

В качестве примера таких отчётов в приложении ReportManager реализован отчёт «Баланс».

Отчёт «Баланс».

Данный отчёт выводит информацию о суммах денежных средств (об остатках денежных средств) находящихся на всех бухгалтерских счетах организации по состоянию на определённую дату.

Интерфейс отчёта представлен на рисунках 5.1 и 5.2. В оконной форме используется интерфейс с двумя закладками: «параметры», «выборка». Данный интерфейс предполагается стандартом для реализации всей группы управленческих, бухгалтерских и статистических отчётов.

Отчёт «Баланс» содержит две закладки:

– первая закладка «параметры» предназначена для ввода информации для выборки данных, в данном случае – для выбора и ввода даты баланса, а также для выбора формы сохранения полученной информации – текст, html-документ или форма табличного процессора Excel. Все полученные отчёты сохраняются в подпапке «Documents» папки «ReportManager»;

- вторая закладка «выборка» отображает полученную по нажатию кнопки «выбрать» информацию – подпрограмма-отчёт переключается на эту закладку сразу после получения данных.

Кроме того, форма содержит:

* верхнюю панель, которая отображает название отчёт (или какую-то дополнительную вспомогательную информацию во время работы отчёта);
* нижняя панель содержит кнопки выборки и сохранения информации н кнопку выхода из подпрограммы;

Рисунок 5.1

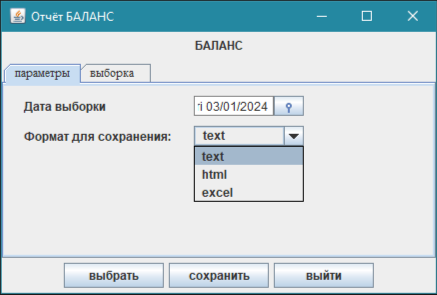
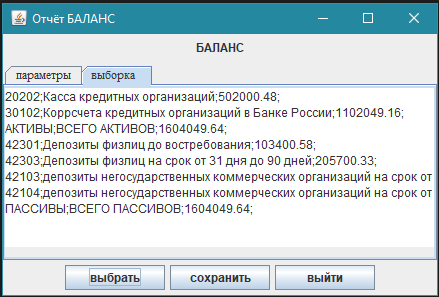


Рисунок 5.2



Реализация отчёта

Подпрограмма-отчёт реализована в виде исполняемого jar-файла с названием balanceReport.jar. Как и все подпрограммы отчёты располагается в подпапке «Reports» папки «ReportManager». Расположение папок хранится в конфигурационном файле Config.json.

UML-диаграмма подпрограммы приведена в приложении №2 страница 53.

Подпрограмма-отчёт организационно разбита на четыре пакета: «model», «view», «presenter» и «repository».

**Пакет «model»** содержит два класса:

* класс «ReportSaver» - предназначен для сохранения полученной из БД информации. В классе реализовано сохранение выборки в текстовый документ, в html-страницу и в файл табличного процессора Excel;
* класс ConfigData – содержит информацию для подключения к базе данных, по расположению рабочих папок приложения ReportManager – рабочая директория приложения, директория для сохранения полученных отчётов и т.д., информация для этого класса подгружается из файла Config.json. Расположение и название файла конфигурации передаётся параметром из основного модуля приложения ReportManager.jar во время вызова подпрограммы balanceReport.jar на исполнение;

**Пакет «view»** это оконная форма для отображения данных введённых пользователем или полученных из презентера, а также для передачи введённых пользователем данных (дата выборки, то есть отчёт «Баланс» за указанную дату) в презентер для дальнейшей обработки.

В пакете «view» для обеспечения архитектурной границы введен интерфейс:

* «iViewBalanceReport» - этот интерфейс имплементируется классом «ViewBalanceReport», интерфейс содержит два метода: getData и sendData – эти методы передают в презентер информацию, введённую пользователем (sendData) на закладке «параметры» или размещают (getData) информацию полученную от презентера в форме на закладке «выборка»;

**Пакет «repository»** обеспечивает связь с базой данных. Содержит интерфейс для обеспечения архитектурной границы и класс реализующий этот интерфейс. Интерфейс «iRepo» содержит методы: setConnectDB, loadInfo. Интерфейс имплементируется классом «DataBase», который работает с информацией используя средства JDBC (Java DataBase Connectivity). Это стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакетов java.sql и javax.sql, входящих в состав Java SE (Standard Edition). Данный стандарт описан спецификацией "[JSR 221 JDBC 4.1 API](http://download.oracle.com/otn-pub/jcp/jdbc-4_1-mrel-spec/jdbc4.1-fr-spec.pdf)". Данная спецификация говорит о том, что JDBC API предоставляет программный доступ к реляционным базам данных из программ, написанных на Java. Так же рассказывает о том, что JDBC API является частью платформы Java и входит поэтому в Java SE и Java EE. JDBC API представлен двумя пакетами: java.sql and javax.sql.

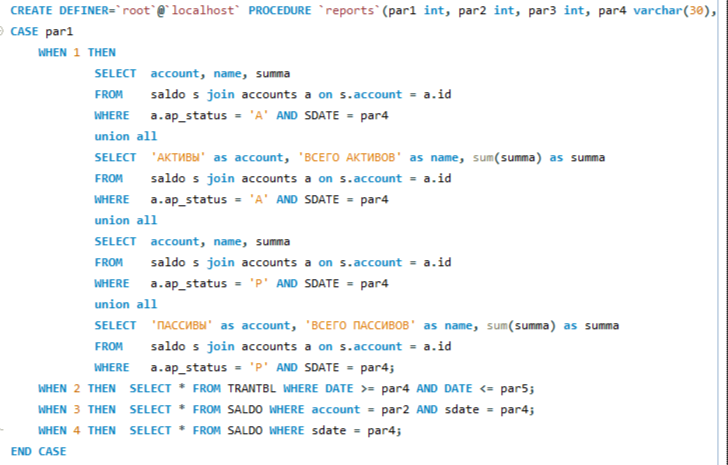
Для действующего (не демонстрационного) приложения ReportManager предполагается широко использовать средства СУБД MySQL, что должно придать дополнительную гибкость при реализации и модификации отчётов. Конкретно здесь речь идёт о использовании такого инструмента СУБД как хранимая процедура («Stored procedure»). Так для каждого отчёта предполагается создать отдельную хранимую процедуру с параметрами, которые будут передаваться из отчёта. Таким образом можно реализовать даже несколько различных выборок для одного отчёта манипулируя параметрами хранимой процедуры.

В рамках представленного приложения ReportManager реализован упрощённый вариант - для выборки используется одна хранимая процедура «reports» с пятью параметрами, где первый параметр обозначает определённый отчёт. Так для рассматриваемой подпрограммы-отчёта вызов хранимой процедуры будет выглядеть следующим образом:

CALL reports(1,0,0,'2024-03-01','');

Первые три параметра имеют тип INT, последние два - VARCHAR. Как видно из приведённого примера здесь для получения баланса достаточно передать номер отчёта – первый параметр и дату выборки.

Хранимая процедура «reports» выглядит следующим образом:



Результаты работы подпрограммы-отчёта «Баланс» представлены на рисунках 5.3, 5.4, 5.5.

**Пакет «presenter»** содержит интерфейс для реализации архитектурной границы. Для использования методов из класса-презентера в классе пакета «view» объявляется поле с типом интерфейса-презентера. После создания класса из пакета «view» методом SetPresenter инициализируется поле имеющее тип интерфейса-презентера, инициализацию этого поля невозможно сделать в конструкторе, так как для создания презентера требуется уже созданный класс из пакета «view».

Пакет содержит интерфейс «iPresenterBalanceReport», который содержит методы getData и saveData и имплементируется классом «PresenterBalanceReport».

Результат работы подпрограммы-отчёта «Баланс»:

Рисунок 5.3

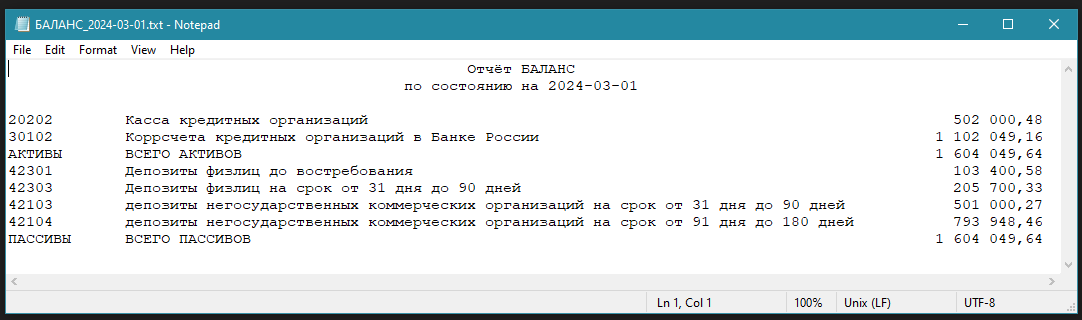
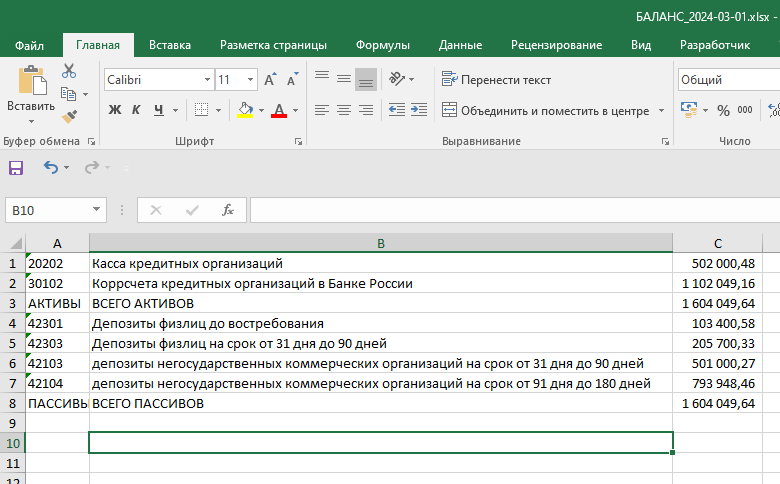


Рисунок 5.4



Рисунок 5.5



3.3.2 Отчёты с предопределённой структурой

Такие отчёты представляют собой установленные госструктурами формы отчётности. Такие формы имеют установленный внешний вид и свою оригинальную структуру размещения информации. Отчёты такого типа несколько сложнее в реализации в плане вывода полученной из БД информации. Опираясь на свой опыт работы с такой отчётностью могу сказать, что удобнее всего работать с Excel-файлами – в файле создаётся шаблон нужной формы, после получения информации из БД файл шаблона заполняется данными и сохраняется с новым именем. Все остальные моменты не отличаются от отчётов с произвольной структурой.

**Глава** 4.

4. Установка программы и вспомогательная информация.

4.1. Установка программы.

Установка программы рассматривается под операционную систему Windows, как наиболее распространённую ОС. Установка состоит из двух этапов:

1. Запуск базы данных. Образ базы данных размещен на сайте по адресу:

*https://hub.docker.com*

Образ для развёртывания контейнера с БД называется vvmarinin/bank:5.0

Для развёртывания под Windows используется приложение «docker desktop». Внешний порт указывается стандартный для MySQL – 3306, если на машине, где поднимается контейнер стандартный порт занят, то необходимо в «docker desktop» в поле «Ports» для «Host port» указать свободный порт и этот же порт должен быть прописан в файле конфигурации Config.info в ключе «ConfigInfoDB». Для корректной работы также нужно заполнить поле «Environment variables», указываем переменную «MYSQL\_ROOT\_PASSWORD» и какой-нибудь пароль для этой переменной – с этим паролем можно будет подключиться к базе данных используя «docker desktop» закладка «Exec».

2. После того, как база данных запущена, можно запускать приложение ReportManager – для Windows в папке «ReportManager» для запуска программы создан файл «ReportManager.bat». Однако предварительно необходимо скачать файлы приложения с ресурса:

*https://github.com/Valentin-V-Marinin/DiplomReportManager.git*

Вся установка заключается в размещении скопированной папки «ReportManager» со всем её содержимым в папку «C:\GB», то есть путь до файла запуска «ReportManager.bat» будет выглядеть так:

C:\GB\ReportManager\ReportManager.bat

Разумеется, папка «ReportManager» может быть помещена и в любое другое место, но тогда необходимо соответствующим образом откорректировать ключи файла конфигурации Config.json, находящегося в папке «ReportManager».

Кроме того, на машине, где разворачивается приложение ReportManager, для работы java-программ должна быть установлена Java не ниже 17 версии.

Развёртывание приложение на операционных системах семейства Linux (тестировалось на Ubuntu 22.04) мало чем отличается от развёртывания на Windows – папка с приложением копируется в домашнюю директорию – в моём случае это была директория «/home/vvm», далее нужно заменить файл конфигурации – где пути прописаны с учётом ОС Linux, образец файла конфигурации (ConfigUbuntu.json) выложен на ресурсе:

*https://github.com/Valentin-V-Marinin/DiplomReportManager.git*

Команда для запуска приложения из папки «/home/vvm/ReportManager» будет выглядеть так:

java –jar ReportManager.jar Config.json

Кроме того, нужно изменить доступ для файлов и папок, для этого использовалась команда:

chmod 777 <имя файла или папки>

4.2. Вспомогательная информация.

При запуске программы необходимо ввести учётные данные, в базе данных заведено для демонстрационных целей два пользователя:

* Маринин Валентин Владимирович

Логин – vvm, пароль – 958

* Вавилова Александра Николаевна

Логин – lady, пароль – lady

В базе данных также введён перечень отчётов, их можно увидеть через подпрограмму-отчёт «АДМИНИСТРАТИВНЫЙ\_ОТЧЁТ» по нажатию кнопки «загрузить» или в подпрограмме-отчёте «АДМИНИСТРАТИВНЫЙ\_ПОЛЬЗОВАТЕЛЬ» на закладке «отчёты» в окне «весь набор отчётов». При этом реально действующие три подпрограммы-отчёта, то есть к названию отчёта прикреплён файл с подпрограммой (jar-файл), перечень действующих отчётов:

* АДМИНИСТРАТИВНЫЙ\_ПОЛЬЗОВАТЕЛЬ
* АДМИНИСТРАТИВНЫЙ\_ОТЧЁТ
* Баланс

Остальные отчёты – пустышки для демонстрационных целей, такой отчёт может быть присвоен (назначен) пользователю, но поскольку к нему не прикреплён файл подпрограммы-отчёта, то при старте программа будет выдавать сообщение, что файл отсутствует в сетевом хранилище. на дальнейшую работоспособность программы это не влияет. При попытке вызова такого отчёта на исполнение ничего не происходит – отчёт не вызывается.

При вызове подпрограмм-отчётов на исполнение – двойной клик по отчёту или кнопка «выбрать» (см. рисунок 3.2 страница 19) возникает пауза от 2 до 6 секунд – в зависимости от мощности компьютера, то есть основное окно гаснет, а окно вызываемой подпрограммы-отчёта появляется через 2-6 секунд.

Для выбора данных в отчёте «Баланс» в базу данных введены данные с остатками на счетах по состоянию на 01.03.2024 (первое марта 2024), соответственно, выбор другой даты в отчёте приведёт к пустому результату.

Внутри папки «ReportManager» находится папка «NetHost». Эта папка имитирует сетевой ресурс на котором по замыслу работы приложения должны располагаться все автоматизированные в рамках приложения ReportManager отчёты организации. Все хосты, на которых установлен ReportManager, во время старта получают список отчётов пользователя, используя его логин, сравнивают hash-число отчётов расположенных на сетевом хосте («NetHost») и на локальном хосте пользователя при несовпадении hash-числа происходит копирование подпрограммы-отчёта с сетевого хоста на локальный ресурс (такая ситуация возникает при модификации старого отчёта или при добавлении пользователя нового отчёта на исполнение).

Вся информация по дипломному проекту находится по адресу:

*https://github.com/Valentin-V-Marinin/Diplom.git*

В том числе:

* папка «DataBase» - sql-скрипты таблиц, триггеров и хранимых процедур использовавшихся для создания базы данных «enterprise»;
* папки «ReportManager», «ReportUser», «Reports», «reportAdmin» - исходный код подпрограмм-отчётов и основного модуля программы ReportManager;
* папка «Схемы» - все графики, рисунки и схемы использовавшиеся в дипломном проекте;
* файл «Diplom.docx» - дипломный проект;

**Заключение**

Как известно, успех любого бизнеса зависит от своевременных и правильных бизнес-решений. А принятию таких решений очень способствует нужный объём и качество информации на основе которой и будет приниматься бизнес-решение. Поэтому роль отчётности очень важна в работе любой организации, её нельзя недооценивать. Недооценка приведёт к снижению конкурентоспособности организации и в конечном счёте - к сворачиванию бизнеса. Таким образом, необходимо постоянно совершенствовать формы отчётности и способы её получения. Необходимо постоянно совершенствовать программное обеспечение по работе с отчётностью.

Поскольку любой успешный бизнес имеет тенденцию к расширению деятельности, то основным путём развития и совершенствования приложения ReportManager видится добавление web-компоненты в приложение, то есть возможность собирать информацию не только из локальной базы данных, но и с помощью web из удалённых баз данных и формирование как консолидированной отчётности по всей организации, так и по каждому филиалу в отдельности – тут, конечно, многое зависит от того, как организована база данных организации. Если агрегирование информации осуществляет основной банковский софт, предназначенный для осуществления транзакций и прочих ежедневных операций, то тогда вся необходимая информация может быть получена из локальной БД.

Desktop-приложение - хорошее решение для небольших банков или предприятий, но если у банка есть филиальная сеть, то масштабирование приложения в варианте «desktop» может быть проблематичным. Кроме того, учитывая, что desktop-приложения отживают своё и есть устойчивая тенденция на развитие web-приложений, возможно, будет актуально создать web-версию приложения ReportManager.

**Список литературы**

Бондарев В. М.

«Учебное пособие по программированию на Java»

Харьков « СМИТ», 2003, 296 с.

Канель Е. Г., Фрайман З.

«Основы программирования на Java: Для школьников … и не только»

М.: ЛЕНАНД, 2019, 200 с.

Мамаев Е. В.

«Microsoft SQL Server 2000»

СПб.: БХВ-Петербург, 2001, 1280 с,

Маслов В. В.

«Основы программирования на языке Java»

М.: Горячая линия – Телеком, 200, 132 с.

Машнин Т. С.

«Современные Java технологии на практике»

СПб.: БХВ-Петербург, 2010, 560 с.

Прохорёнок Н. А.

«Основы Java»

СПб.: БХВ-Петербург, 2017, 704 с,

Тихомиров Ю. В.

«MS SQL Server 2000: разработка приложений»

СПб.: БХВ-Петербург, 2000, 368 с,

Брайсон Пейн

«Легкий способ выучить Java»

М.: Эксмо, 2019, 400 с.

Брюс Эккель

«Философия Java»

СПб.: Питер, 2009, 630 с,

Герберт Шилдт

«Java: Руководство для начинающих»

М.: ООО «И. Д. Вильямс», 2012, 624 с.

Герберт Шилдт

«Java. Полное руководство»

М.: ООО «Альфа-книга», 2018, 1488 с.

Кен Коузен

«Современный Java/ Рецепты программирования»

М.: ДМК Пресс, 2018, 274 с.

Скотт Кёрк

«Java для студента»

СПб.: БХВ-Петербург, 2007, 448 с,

Интернет ресурсы:

*https://devmark.ru/article/md5-hashing*

Вычисление MD5-хэша

*https://ru.wikipedia.org/wiki/Хеш-функция*

Хеш-функция

*https://ru.wikipedia.org/wiki/UML*

UML (*Unified Modeling Language*)

*https://dev.to/sumana2001/build-a-custom-mysql-docker-container-404f*

Sumana Basu, Build a custom MySQL Docker Container

**Приложения**

1. Приложение №1 – UML-диаграмма подпрограммы-отчёта «АДМИНИСТРАТИВНЫЙ\_ПОЛЬЗОАВАТИЕЛЬ» см. стр. 21.

Также UML-диаграмма размещена на ресурсе:

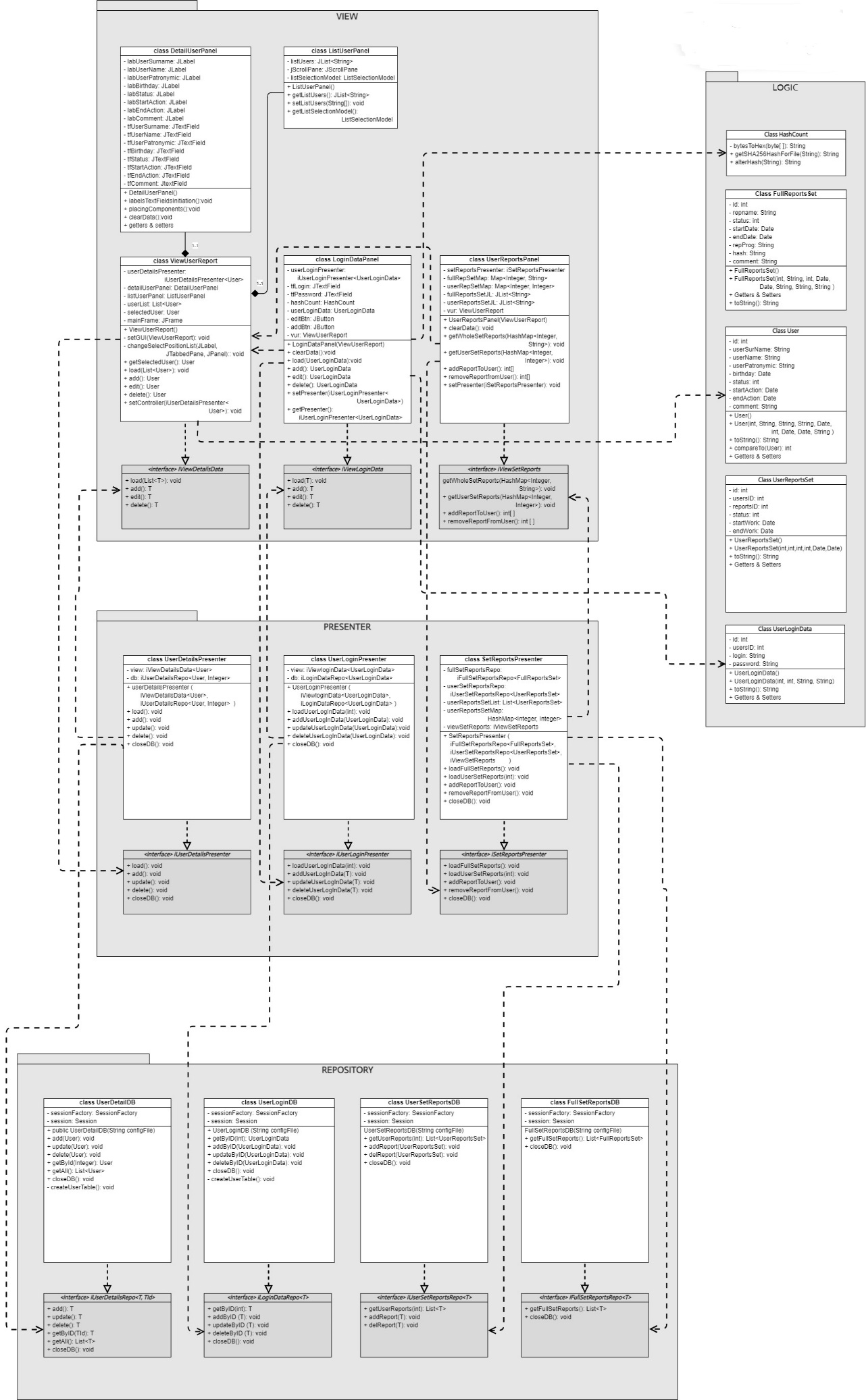
https://github.com/Valentin-V-Marinin/Diplom.git

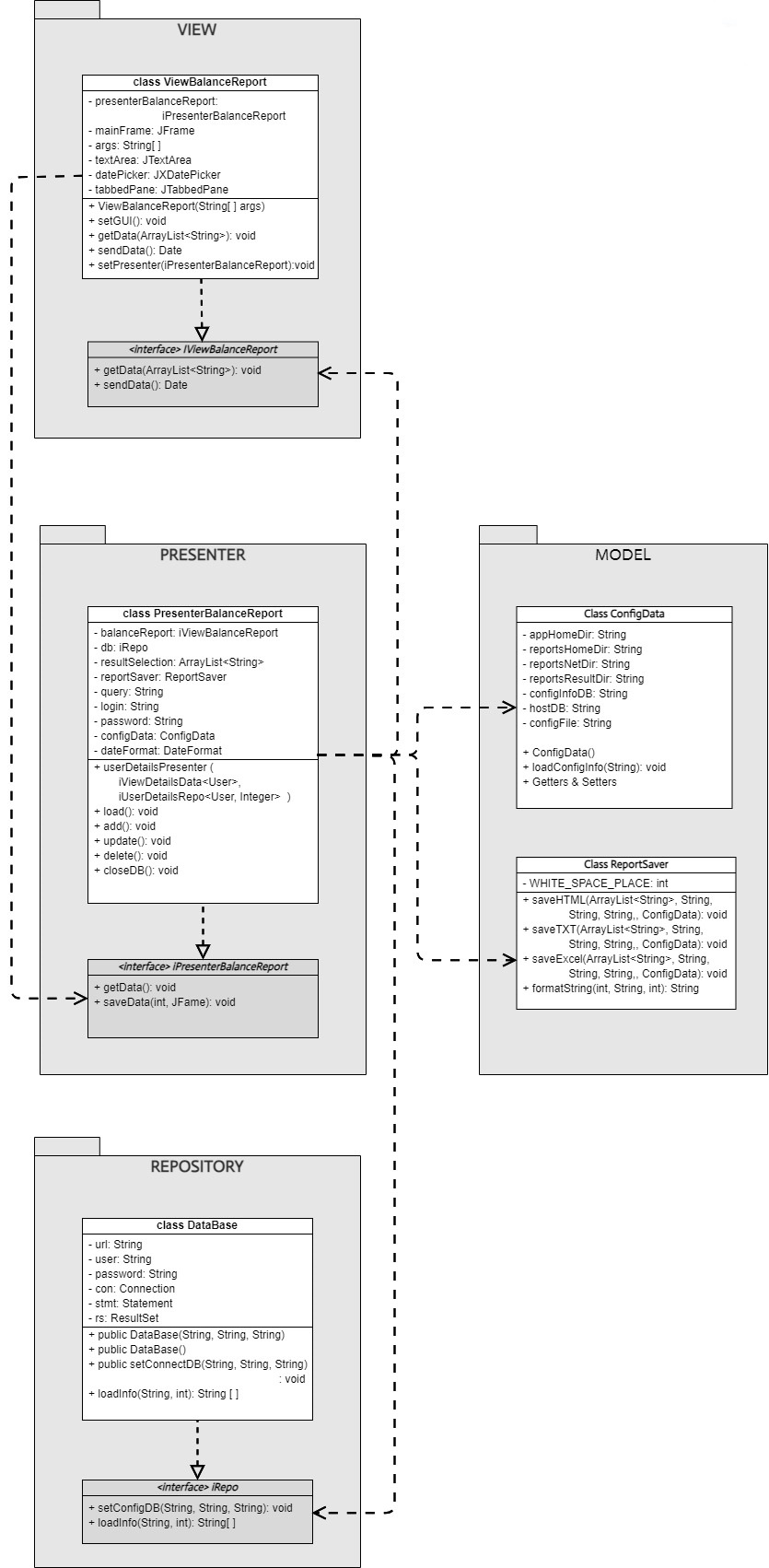
папка «Схемы», файл «reportUser.jpg»

1. Приложение №2 – UML-диаграмма подпрограммы-отчета «Баланс» см. стр. 38.

https://github.com/Valentin-V-Marinin/Diplom.git

папка «Схемы», файл «balanceReport.jpg»

****

****